Software Engineering Internship – Assignment Report

Project: Library Management System
Name: Kalmi Dilara
Date: 16 February 2026

Introduction

The Library Management System is a web-based application designed to manage books in a library efficiently. The main goal of this project is to allow users to create, view, update, and delete book records in a simple, user-friendly interface.

Technologies used:

- Backend: C# .NET Web API, Entity Framework Core, SQLite
- Frontend: React with TypeScript
- Tools: VS Code, Git

System Architecture

The system follows a client-server architecture:

React Frontend → HTTP Requests → .NET API → Entity Framework → SQLite Database

Backend Implementation

Technologies Used:

- C# .NET Web API
- Entity Framework Core
- SQLite database
- RESTful API design

Project Structure:

- Controllers: BookController handles all CRUD operations.
- Models: Book model contains Id, Title, Author, Description.
- DbContext: LibraryContext manages database access.

API Endpoints:

Method	Endpoint	Description
GET	/api/book	Retrieve all books
GET	/api/book/{id}	Retrieve a single book
POST	/api/book	Add a new book

Method	Endpoint	Description
PUT	/api/book/{id}	Update an existing book
DELETE	/api/book/{id}	Delete a book

Error Handling:

- Requests validated for required fields.
- BadRequest for invalid input.
- NotFound for missing records.

Frontend Implementation

Technologies Used:

- React with TypeScript
- Axios for HTTP requests
- React Router for navigation

Component Structure:

- App.tsx: Main component.
- BookList.tsx: Displays list of books.
- BookForm.tsx: Handles create and update operations.

CRUD Operations:

- Create: Add new book with title, author, description.
- Read: View all books.
- Update: Edit existing books.
- Delete: Remove selected books.

Input Validation:

- Forms validate empty fields.
- Error messages displayed for invalid input.

Database Design

SQLite chosen for simplicity.
Book Table Structure:

Field	Type
Id	int PK
Title	string
Author	string
Description	string

Integration Between Frontend & Backend

- Frontend communicates via HTTP requests.
- Tested in browser for seamless functionality

Challenges Faced

- State Management Issues: Correct use of useEffect for UI refresh.
- Database Conflicts: Careful migration handling.

Additional Features

- Responsive UI design.
- Confirmation dialogs for deletion.
- Toast notifications for successful operations.

How to Run the Application

Backend:
cd Backend
dotnet restore
dotnet run

- .NET version: 10.0.103
- API runs at: http://localhost:3000

Frontend:
cd frontend

npm install
npm run dev

- Node version: v25.2.1
- Environment variable: VITE_BASE_API_URL=http://localhost:3000/api/book
- For full source code, please visit the GitHub repository: https://github.com/Kalmid/Library-MS.git

Key Learnings & Reflection

- Developed understanding of RESTful API design.
- Learned React frontend integration with C# backend.
- Gained experience with SQLite and Entity Framework.
- Improved error handling, validation, and user experience.
- Learned importance of clean code, documentation, and Git version control.